MACHINE LEARNING LAB

Topic: Bank Customer Churn Prediction

Dataset:

https://www.kaggle.com/datasets/shantanudhakadd/bank-customer-churn-prediction

Team:

1. Srushti Dhakate (25)
2. Prathamesh Gujar (52)
3. Prathamesh Rajbhoj (53)

Importing Libraries

```
import pandas as pd
import seaborn as sns

from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score,recall_score,f1_score

from sklearn.linear_model import LogisticRegression
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier

from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

```
data = pd.read_csv('Churn_Modelling.csv')
```

Display Top 5 Rows of The Dataset

```
data.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1

Check Last 5 Rows of The Dataset

```
data.tail()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMemb
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	

Data Info

```
data.shape
```

```
(10000, 14)
```

```
print("Number of Instances", data.shape[0])
print("Number of Attributes", data.shape[1])
```

```
Number of Instances 10000
Number of Attributes 14
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  int64  
 4   Geography        10000 non-null  object 
 5   Gender           10000 non-null  object 
 6   Age              10000 non-null  int64  
 7   Tenure           10000 non-null  int64  
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64  
 10  HasCrCard        10000 non-null  int64  
 11  IsActiveMember   10000 non-null  int64  
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

Check Null Values In The Dataset

```
data.isnull().sum()
```

```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

Statistics About The Dataset

```
data.describe()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balar
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000

Dropping Columns

```
data.columns
```

```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090

```
data = data.drop(['RowNumber', 'CustomerId', 'Surname'], axis=1)
```

```
data.head()
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	I
0	619	France	Female	42	2	0.00	1	1	
1	608	Spain	Female	41	1	83807.86	1	0	
2	502	France	Female	42	8	159660.80	3	1	
3	699	France	Female	39	1	0.00	2	0	
4	850	Spain	Female	43	2	125510.82	1	1	

8. Encoding Categorical Data

```
data['Geography'].unique()
```

```
array(['France', 'Spain', 'Germany'], dtype=object)
```

```
data = pd.get_dummies(data,drop_first=True)
```

```
data.head()
```

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Est
0	619	42	2	0.00	1	1	1	
1	608	41	1	83807.86	1	0	1	
2	502	42	8	159660.80	3	1	0	
3	699	39	1	0.00	2	0	0	
4	850	43	2	125510.82	1	1	1	

Imbalanced Dataset

```
data['Exited'].value_counts()
```

```
0    7963
1    2037
Name: Exited, dtype: int64
```

Seperating Input & Result attributes

```
X = data.drop('Exited',axis=1)
y = data['Exited']
```

Handling Imbalanced Data With SMOTE

```
X_res, y_res = SMOTE().fit_resample(X, y)
```

```
y_res.value_counts()
```

```
1    7963
0    7963
Name: Exited, dtype: int64
```

Splitting The Dataset Into The Training Set And Test Set

```
X_train, X_test, y_train, y_test = train_test_split(X_res, y_res, test_size=0.20, random_state=42)
```

Feature Scaling

```
sc = StandardScaler()
```

```
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
X_train
```

```
array([[ 1.13939368,  2.32118674, -1.39738675, ...,  1.74778208,
        -0.46075397,  1.18749315],
       [ 1.09559174,  0.1094956 , -1.39738675, ..., -0.57215371,
        -0.46075397,  1.18749315],
       [ 0.20860244, -0.19209865, -1.0296382 , ..., -0.57215371,
        -0.46075397, -0.84211012],
       ...,
       [ 0.19765196,  0.21002701,  1.17685308, ..., -0.57215371,
        -0.46075397,  1.18749315],
       [-1.11640626,  0.00896418,  1.54460163, ...,  1.74778208,
        -0.46075397, -0.84211012],
       [ 0.52616651,  1.01427834, -1.39738675, ...,  1.74778208,
        -0.46075397,  1.18749315]])
```

```
# Model Analysis Formulas

# precision = TP / (FP + TP)
# recall = TP / (TP + FN)
# f1score = (2 * precision * recall) / (precision + recall)
```

KNeighbors Classifier

```
knn = KNeighborsClassifier()
```

```
knn.fit(X_train,y_train)
```

```
y_pred3 = knn.predict(X_test)
```

```
accuracy_score(y_test,y_pred3)
```

```
0.8214061519146265
```

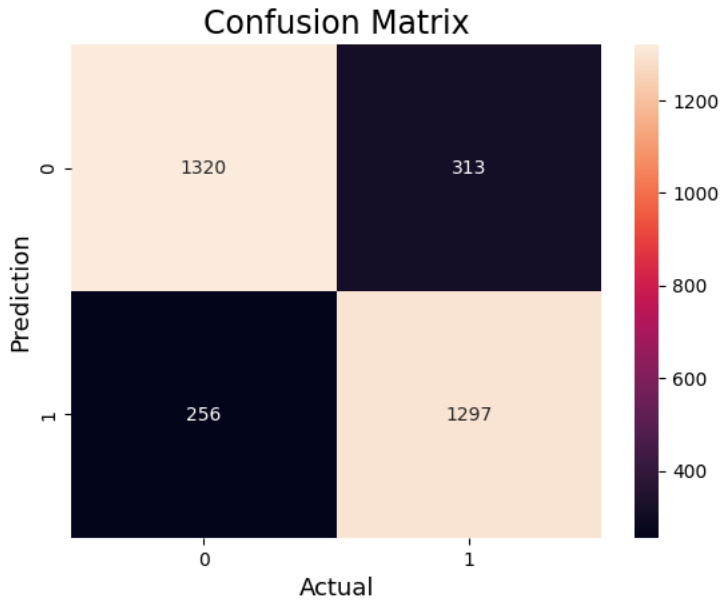
```
cm3 = confusion_matrix(y_test,y_pred3)
# print('Confusion matrix\n\n', cm)

sns.heatmap(cm3,
            annot=True,
            fmt='g',
            xticklabels=['0','1'],
            yticklabels=['0','1'])


plt.ylabel('Prediction',fontsize=13)
plt.xlabel('Actual',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()

print("\n\n\n")

print(classification_report(y_test,y_pred3))
```



	precision	recall	f1-score	support
0	0.84	0.81	0.82	1633
1	0.81	0.84	0.82	1553

Decision Tree Classifier

```
dt = DecisionTreeClassifier()
```

```
dt.fit(X_train,y_train)
```

```
root_node = dt.tree_
# Print information about the root node
print("Feature index for the root node:", root_node.feature[0])
print("Threshold value for the root node:", root_node.threshold[0])
```

```
Feature index for the root node: 1
Threshold value for the root node: -0.04130152519792318
```

```
cols = data.drop('Exited',axis=1).columns
feature_names = np.array(cols)
```

```
# Access the root node's feature index
root_feature_index = dt.tree_.feature[0]

# Get the feature name for the root node
root_feature_name = feature_names[root_feature_index]

# Print the feature name of the root node
print("Feature name for the root node:", root_feature_name)
```

```
Feature name for the root node: Age
```

```
y_pred4 = dt.predict(X_test)
```

```
accuracy_score(y_test,y_pred4)
```

```
0.7937853107344632
```

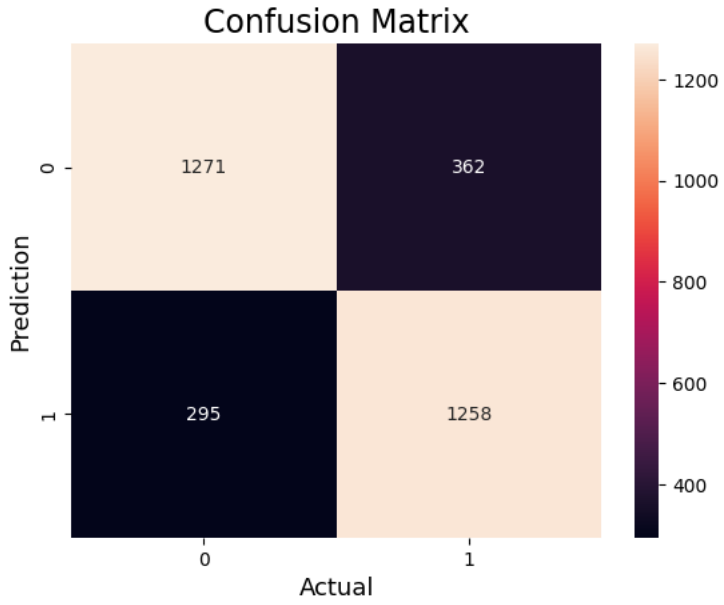
```
cm4 = confusion_matrix(y_test,y_pred4)
# print('Confusion matrix\n\n', cm)

sns.heatmap(cm4,
            annot=True,
            fmt='g',
            xticklabels=['0','1'],
            yticklabels=['0','1'])
```

```
plt.ylabel('Prediction',fontsize=13)
plt.xlabel('Actual',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()

print("\n\n\n")

print(classification_report(y_test,y_pred4))
```




```
              precision    recall  f1-score   support

           0       0.81      0.78      0.79      1633
           1       0.78      0.81      0.79      1553

    accuracy                           0.79      3186
   macro avg       0.79      0.79      0.79      3186
weighted avg       0.79      0.79      0.79      3186
```

Random Forest Classifier

```
rf = RandomForestClassifier()
```

```
rf.fit(X_train,y_train)
```

```
RandomForestClassifier()
```

```
y_pred5 = rf.predict(X_test)
```

```
accuracy_score(y_test,y_pred5)
```

```
0.8637790332705587
```

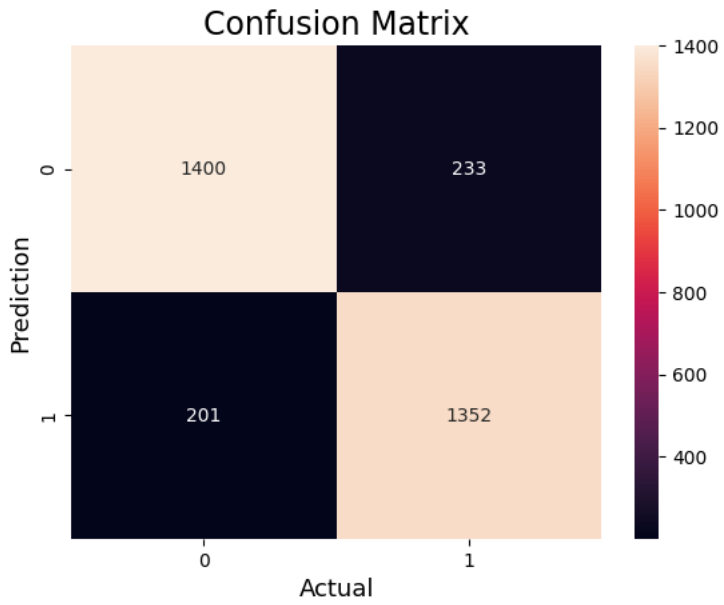
```
cm5 = confusion_matrix(y_test,y_pred5)
# print('Confusion matrix\n\n', cm)

sns.heatmap(cm5,
            annot=True,
            fmt='g',
            xticklabels=['0','1'],
            yticklabels=['0','1'])


plt.ylabel('Prediction',fontsize=13)
plt.xlabel('Actual',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()
```

```
print("\n\n\n")

print(classification_report(y_test,y_pred5))
```




```
              precision    recall  f1-score   support

           0       0.87      0.86      0.87      1633
           1       0.85      0.87      0.86      1553

    accuracy                           0.86      3186
   macro avg       0.86      0.86      0.86      3186
weighted avg       0.86      0.86      0.86      3186
```

Model Comparision

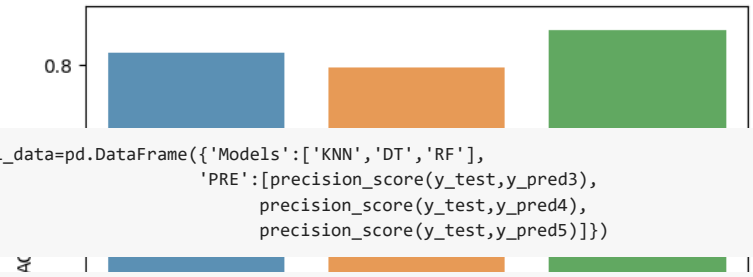
```
final_data=pd.DataFrame({'Models':['KNN','DT','RF'],
                         'ACC':[accuracy_score(y_test,y_pred3),
                                accuracy_score(y_test,y_pred4),
                                accuracy_score(y_test,y_pred5)]})
```

```
final_data
```

	Models	ACC
0	KNN	0.821406
1	DT	0.793785
2	RF	0.863779

```
sns.barplot(x=final_data['Models'], y=final_data['ACC'], alpha=0.8)
```

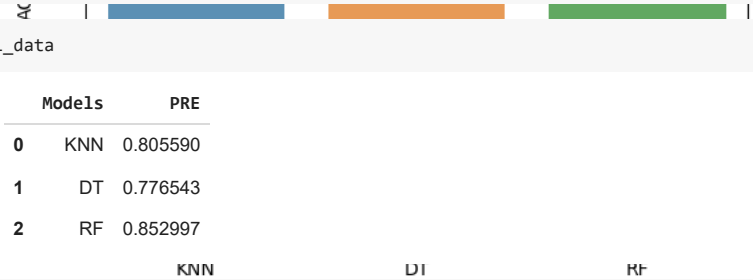
```
<Axes: xlabel='Models', ylabel='ACC'>
```




```
final_data=pd.DataFrame({'Models':['KNN','DT','RF'],
                         'PRE':[precision_score(y_test,y_pred3),
                                precision_score(y_test,y_pred4),
                                precision_score(y_test,y_pred5)]})
```

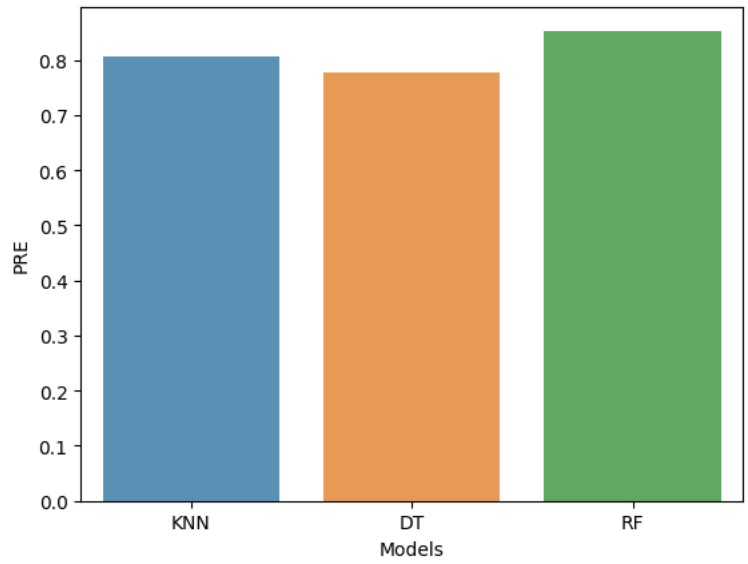
```
final_data
```

	Models	PRE
0	KNN	0.805590
1	DT	0.776543
2	RF	0.852997




```
sns.barplot(x=final_data['Models'], y=final_data['PRE'], alpha=0.8)
```

```
<Axes: xlabel='Models', ylabel='PRE'>
```



18. Save The Model

```
X_res=sc.fit_transform(X_res)
```

```
rf.fit(X_res,y_res)
```

```
RandomForestClassifier
RandomForestClassifier()
```

```
import joblib
```

```
joblib.dump(rf,'churn_predict_model')
```

```
['churn_predict_model']
```

```
model = joblib.load('churn_predict_model')
```

```
data.drop('Exited',axis=1).columns
```

```
Index(['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Geography_Germany',
       'Geography_Spain', 'Gender_Male'],
      dtype='object')
```

```
prediction = model.predict([
    [
```

```
        619,
        42,
        2,
        1000.0,
        0,
        0,
        0,
        101348.88,
        0,
        0,
        0]
  ])



# prediction = model.predict([
#     [
#         619,
#         50,
#         1,
#         150000.0,
#         0,
#         0,
#         0,
#         101348.88,
#         0,
#         0,
#         1]
#   ])

prediction
```

```
array([1])
```

```
output = ["Customer will leave", "Customer will continue"]

print(output[prediction[0]])
```

```
Customer will leave
```